Harbor私有镜像仓库

[liangdong@smzdm.com](mailto:liangdong@smzdm.com)

[Harbor介绍 1](#_Toc531271868)

[环境准备 2](#_Toc531271869)

[安装依赖 2](#_Toc531271870)

[搭建harbor 3](#_Toc531271871)

[配置harbor 3](#_Toc531271872)

[首次安装 4](#_Toc531271873)

[配置nginx反向代理 4](#_Toc531271874)

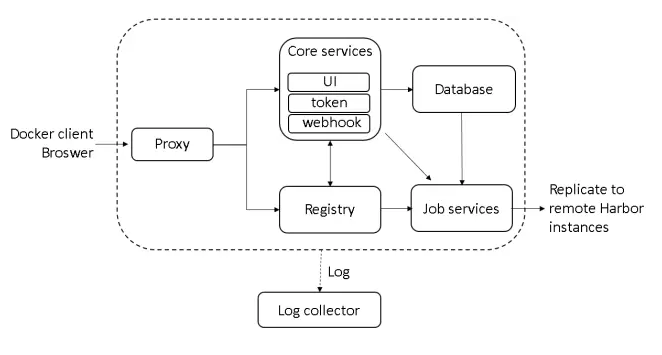
[添加项目 5](#_Toc531271875)

[上传镜像 6](#_Toc531271876)

[镜像同步 9](#_Toc531271877)

[故障转移 11](#_Toc531271878)

# Harbor介绍



参考： <https://github.com/goharbor/harbor/wiki/Architecture-Overview-of-Harbor>

Harbor使用docker官方的Registry仓库方案保存镜像，使用数据库保存harbo的一些配置信息。

Harbor默认把所有这一套组件全部放在容器里跑，至于如何做高可用和扩展性呢？ 官方目前只提供2种方法：

1. 高难度方案
   1. 配置使用分布式storage，比如ceph，这样就可以搭建多个harbor共享一套镜像文件了。
   2. 数据库独立搭建postgresql主从，这样搭建多个harbor就可以共享一套配置信息了。
2. 弱鸡方案
   1. 单机磁盘存储镜像文件，然后搭建多个harbor，让其中1个harbor接收上传镜像，然后再向其他harbor同步镜像，这是harbor的一个复制功能。
   2. 数据库使用自带的容器版本，这样运维改配置类的东西必须在所有harbor中操作一次。（注意，弱鸡方案中，不能让多个harbor共享一个数据库，因为复制配置也会被多个harbor共享，比如在A harbor配置复制镜像给B，结果B也读到了这个配置，自己复制给自己？就乱套了。）

综合这个现状，我提出一个可行的弱鸡方案。

1，搭建1个harbor主节点，全部使用harbor单机容器，定时备份postgresql数据库。

令域名push.harbar.smzdm.com指向主节点，镜像上传以及WEB UI管理走这个域名。镜像复制给N个harbor从节点。

2，搭建N个harbor从节点，全部使用harbor单机容器，域名pull.harbor.smzdm.com指向主节点+从节点做负载均衡，镜像拉取都走这个域名。

如果主节点宕机，选一个从节点，把备份的postgresql中的registry库导进去，把push.harbor.smzdm.com指过去，然后继续用算恢复了。

# 环境准备

3台ubuntu 16虚拟机，1核1G。

节点1：做harbor主节点

节点2：做harbor从节点

节点3： 部署nginx反向代理，其中push.harbor.smzdm.com域名反向代理到harbor主节点。pull.harbor.smzdm.com负载均衡到2个harbor节点，仅供拉取。

# 安装依赖

新ubuntu没有sshd服务：

apt-get install ssh

安装docker：

**apt-get update**

**apt-get install -y docker.io**

安装python：

apt-get install python

安装openssl（ubuntu默认带着了）：

apt-get install openssl

安装docker-compose（单机的容器编排工具，和k8s的yaml长的差不多，参考：<https://github.com/docker/compose/releases> ）：

curl -L https://github.com/docker/compose/releases/download/1.23.2/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose

# 搭建harbor

## 配置harbor

下载harbor的**offline installer，目前是v1.6.2版本**：

<https://github.com/goharbor/harbor/releases>

打开节点1与节点2的harbor.cfg。

hostname：改为节点自身IP，两台机器不一样。

customize\_crt：改为off，docker login登录harbor时，harbor会用一对rsa公私钥签一个登录token给用户。因为我们最终有N个个harbor对外提供拉取服务，所以得每个harbor的rsa公私钥是一样的，否则在节点1申请的token在节点2无法认证通过，就很尴尬。设置成off的意思就是使用harbor默认的一套Rsa证书，以实现N个节点相同的目的。

harbor\_admin\_password：管理员密码，内置管理员帐号叫做admin。

auth\_mode：harbor的登录账号使用数据库的，还是用公司ldap的，我们先保持db\_auth，未来可以接入ldap。

project\_creation\_restriction ：改成adminonly，只有管理员帐号可以建新项目。

## 首次安装

执行./install.sh即可全自动安装，postgresql里的表都会由adminserver组件自动创建。

harbor的组件都运行在docker container里，暴露的端口都映射到了宿主机上，持久化的数据也映射在宿主机的/data目录，日志在/var/log/harbor目录（日志会自动滚动，不需要清理）。

现在，我们打开浏览器输入IP地址即可访问到harbor：<http://172.18.10.236>，帐号admin，密码刚才我们配置过，默认是Harbor12345。

我们按照同样的方法配置harbor从节点，注意hostname与第一个节点的配置不同，其他都一样。

## 配置nginx反向代理

我们在节点3上启动一个nginx做反向代理，做2个HOST出来：

1. push.harbor.smzdm.com：给jenkins构建发布系统用的，推镜像到harbor主节点，然后同步到harbor从节点。主节点故障期间只影响上传镜像。
2. pull.harbor.smzdm.com：给k8s集群拉镜像用的，负载均衡所有harbor，具备扩展性。

在节点上3操作。

安装nginx：apt-get install nginx

修改配置文件：

cat /etc/nginx/sites-enabled/default

upstream harbor\_pull {

server 172.18.10.236:80 weight=1;

server 172.18.10.240:80 weight=1;

}

upstream harbor\_push {

server 172.18.10.236:80 weight=1;

#server 172.18.10.240:80 weight=1;

}

server {

listen 80;

server\_name pull.harbor.smzdm.com;

location / {

proxy\_pass http://harbor\_pull;

}

}

server {

listen 80;

server\_name push.harbor.smzdm.com;

location / {

proxy\_pass http://harbor\_push;

}

}

另外需要修改一下上传文件大小的限制，否则镜像太大传不上来：

http {

##

# Basic Settings

##

client\_max\_body\_size 0;

chunked\_transfer\_encoding on;

即在http段加上述2行配置。

重启nginx：systemctl restart nginx

在自己的PC上配置一下/etc/hosts，指向nginx即可：

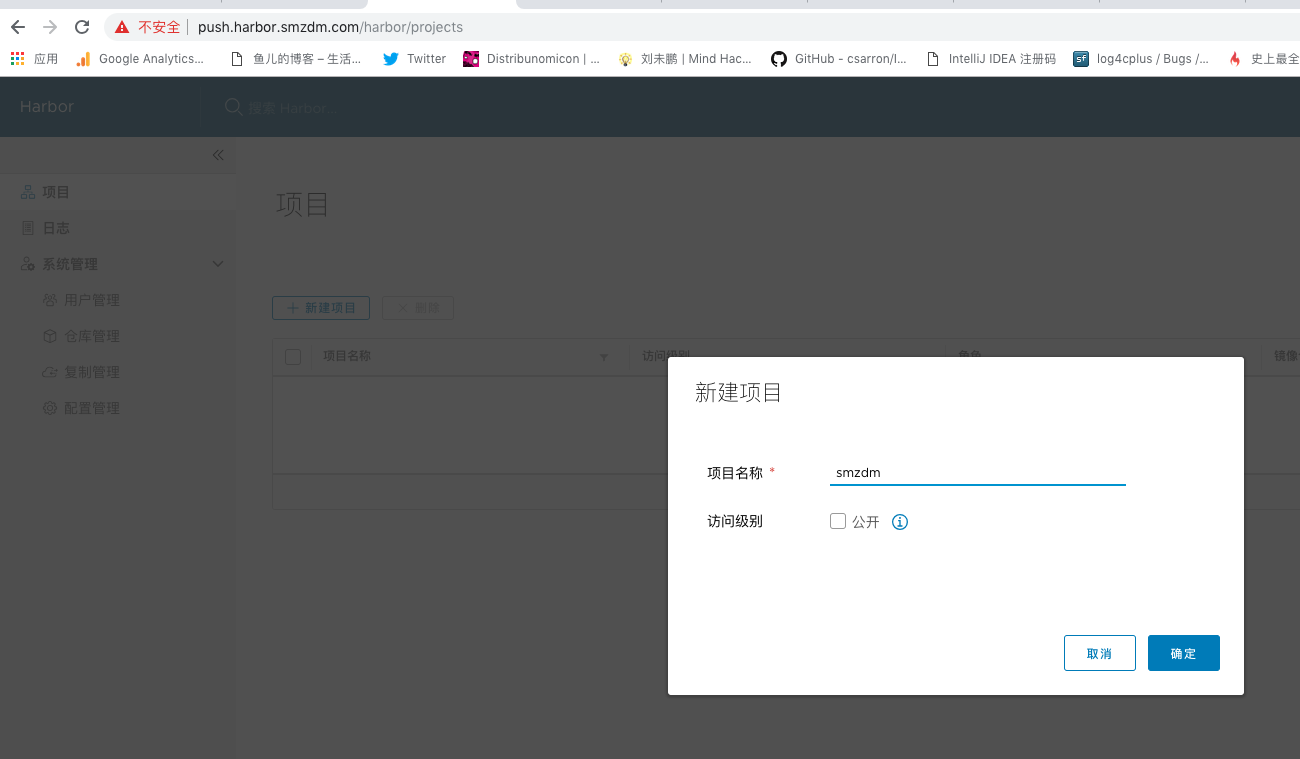
172.18.10.248 push.harbor.smzdm.com

172.18.10.248 pull.harbor.smzdm.com

确认浏览器访问2个域名正常。

## 添加项目

在主harbor上建立smzdm项目：



设置为非公开项目，这意味着如果要pull这个项目下的镜像，需要docker login登录才行。

## 上传镜像

把上述hosts配置到节点3上的/etc/hosts：

172.18.10.248 push.harbor.smzdm.com

172.18.10.248 pull.harbor.smzdm.com

我们在节点3上用docker push来尝试上传一个镜像到harbor仓库。

我们说过push.harbor.smzdm.com指向主节点，镜像将推送到主节点的磁盘上。

因为我们的nginx反向代理没有开始TLS，而docker默认要求仓库是https的，所以需要给节点3上的dockerd配置一个白名单，允许使用http访问push.harbor.smzdm.com和pull.harbor.smzdm.com仓库：

vim /etc/docker/daemon.json

{

"insecure-registries" : ["pull.harbor.smzdm.com:80", "push.harbor.smzdm.com:80"]

}

（最好还是给nginx配上TLS证书，这样就省事了）

现在下载一个Nginx镜像做测试：

docker pull nginx

然后改名到我们的私有仓库地址：

docker tag nginx:latest push.harbor.smzdm.com:80/smzdm/nginx:v1.0

登录到私有仓库：

docker login --username admin --password Harbor12345 push.harbor.smzdm.com:80

docker login --username admin --password Harbor12345 pull.harbor.smzdm.com:80

查看保存的登录密码：

root@harbor01:~/harbor# cat ~/.docker/config.json

{

"auths": {

"pull.harbor.smzdm.com:80": {

"auth": "YWRtaW46SGFyYm9yMTIzNDU="

},

"push.harbor.smzdm.com:80": {

"auth": "YWRtaW46SGFyYm9yMTIzNDU="

}

}

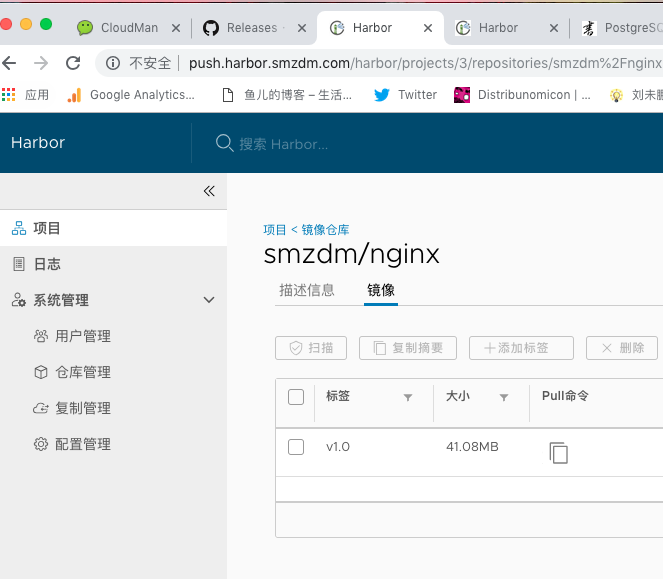
}

以后就免登陆了。

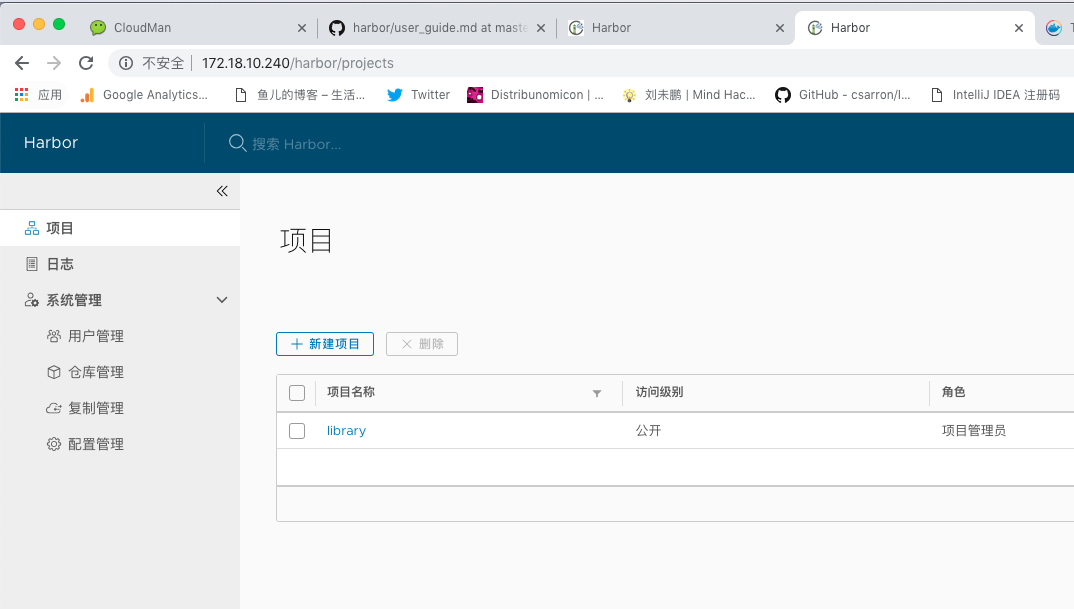
现在就可以push一波了：

docker push push.harbor.smzdm.com:80/smzdm/nginx:v1.0

现在看push.harbor.smzdm.com主节点上，已经有镜像了：



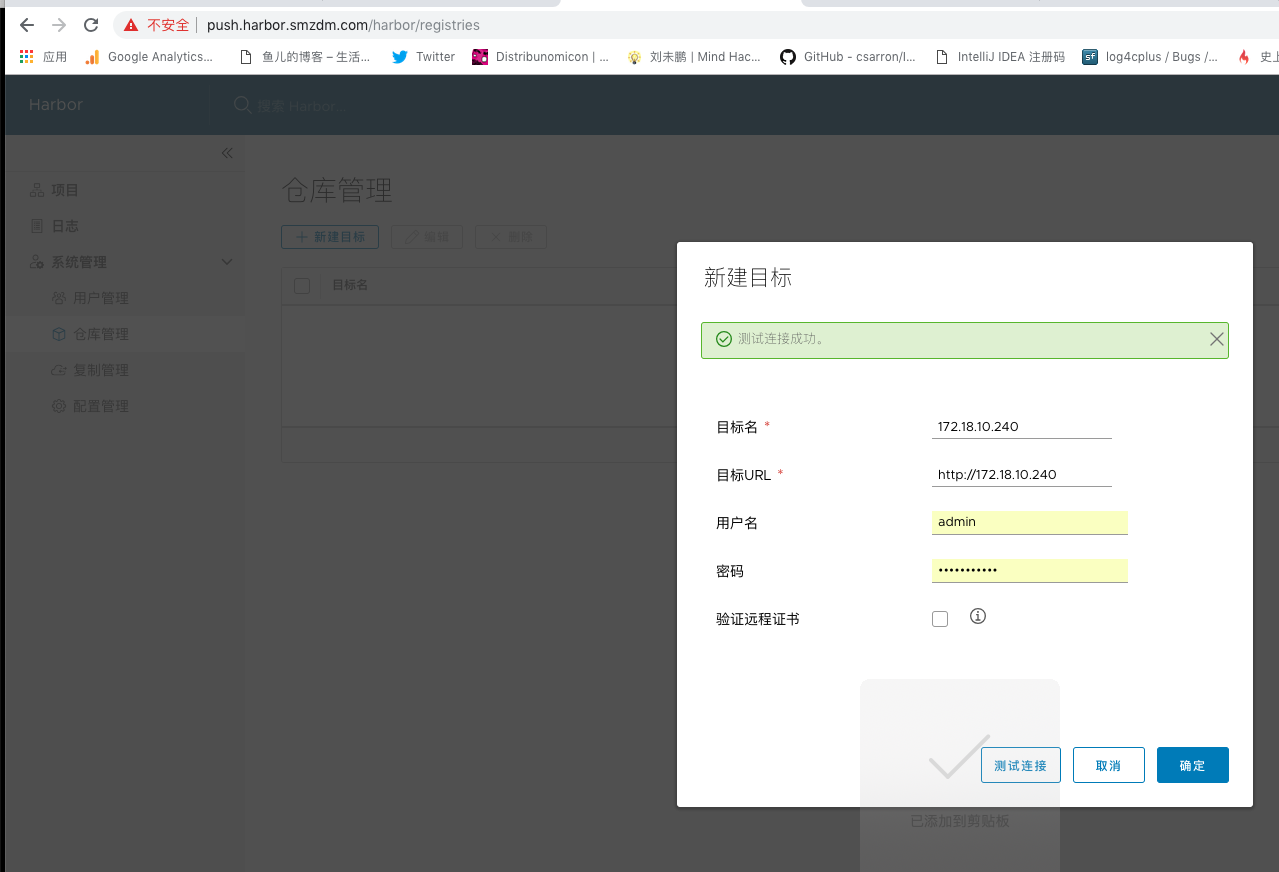
用IP明确的访问harbor从节点，发现没有镜像：



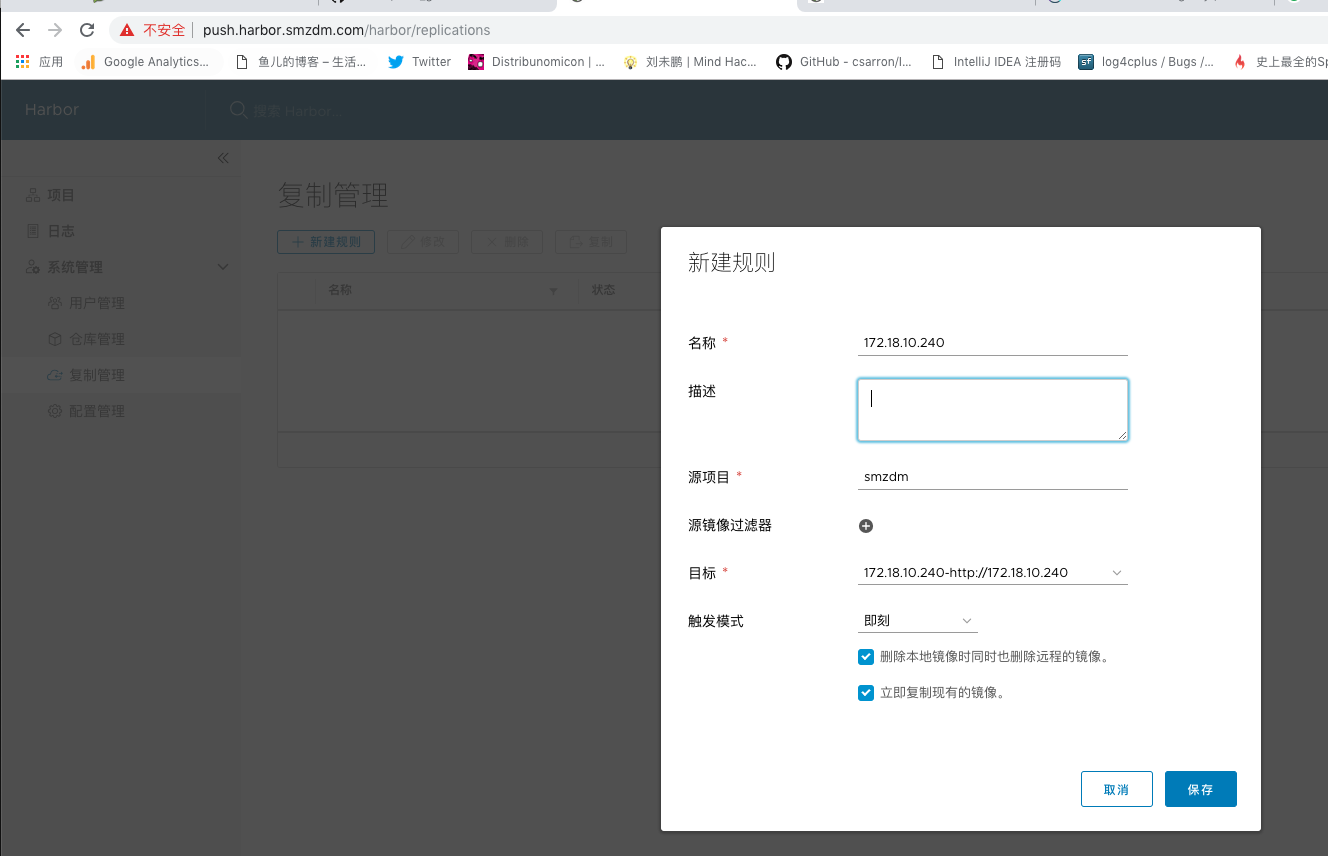
这就是我们接下来要说的，镜像同步。

## 镜像同步

在主harbor上，点击仓库管理，把从harbor配置上来：



新建同步规则：



日志显式复制正常：



查看harbor从库：



也有了。

我们可以pull一下镜像试试：

docker pull pull.harbor.smzdm.com:80/smzdm/nginx:v1.0

注意，拉镜像的时候使用的是pull.harbor.smzdm.com，也就是从任意harbor节点下载镜像。

我们之前推送的时候用的是push.harbor.smzdm.com:80，推送和拉取使用不同的域名没有关系，最终只要让k8s通过URL把镜像拉下来就能执行。

# 故障转移

首先，我们只使用admin帐号完成镜像上传与下载，因为所有harbor初始化过程都会有统一的admin账号和密码保存到各自的postgresql数据库中，这个信息是集群一致的。

其次，我们每个harbor的token服务的rsa公私钥匙一样的，所以docker login拿到的token可以在任意harbor进行认证和鉴权。

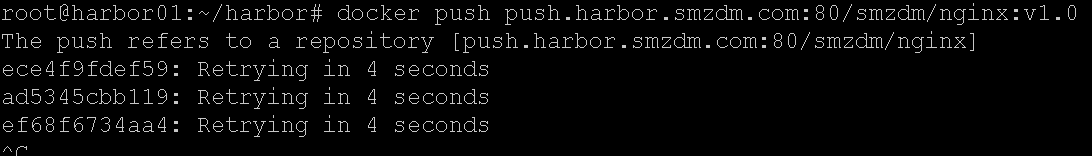
再其次，上传镜像只发给harbor主节点，然后自动复制给所有从节点，所以从任意从节点都可以拉取到完整镜像。

有了这些前提之后，我们可以轻松的实现harbor主节点的故障转移，下面简单模拟一波。

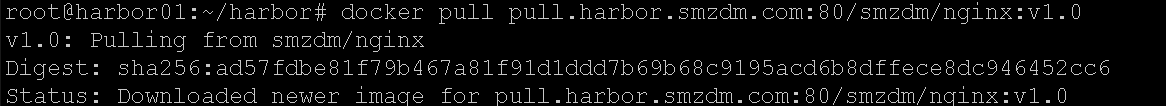
先关掉harbor主节点，假装这台机器挂了：

docker-compose down -v

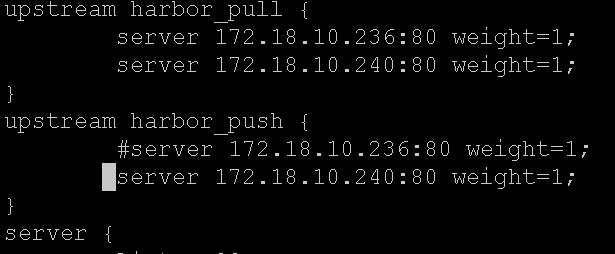
现在尝试push会失败：



尝试pull仍旧会成功，因为harbor从库还活着：

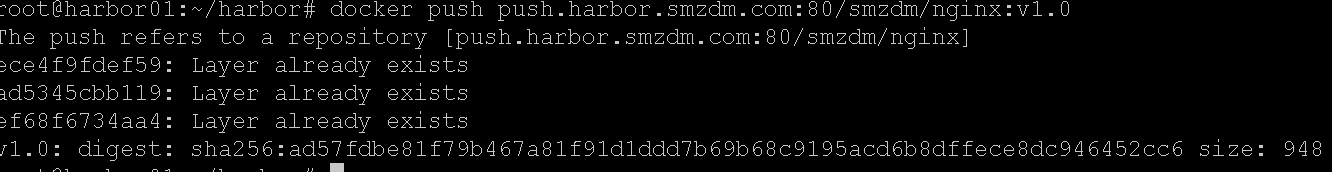


现在只需要找一个harbor从库，在nginx上把push.harbor.smzdm.com指过去：

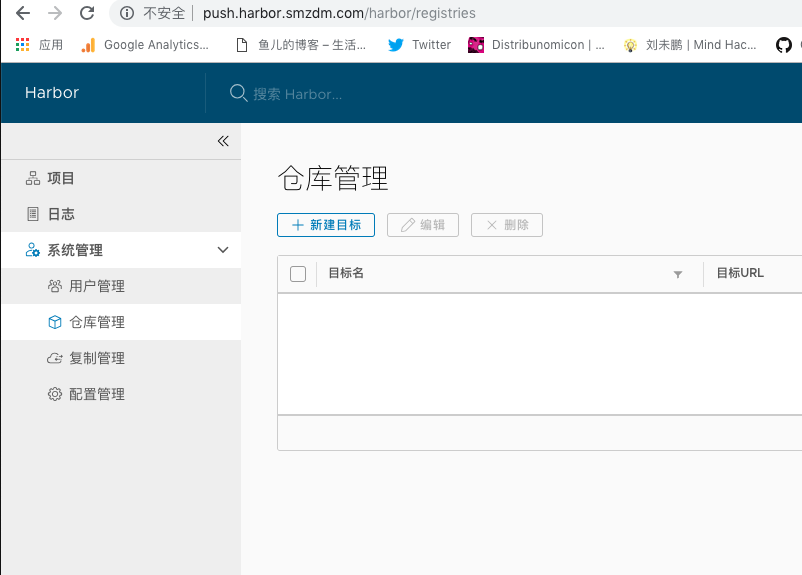


重启nginx：systemctl restart nginx

现在push恢复：



我们harbor主库：



仓库和复制规则都没了，我们现在重新配置一下同步给其他节点即可。

因为我们搭建的集群就2个harbor节点，所以我现在把挂掉的harbor机器重置一下，作为一个新harbor使用。

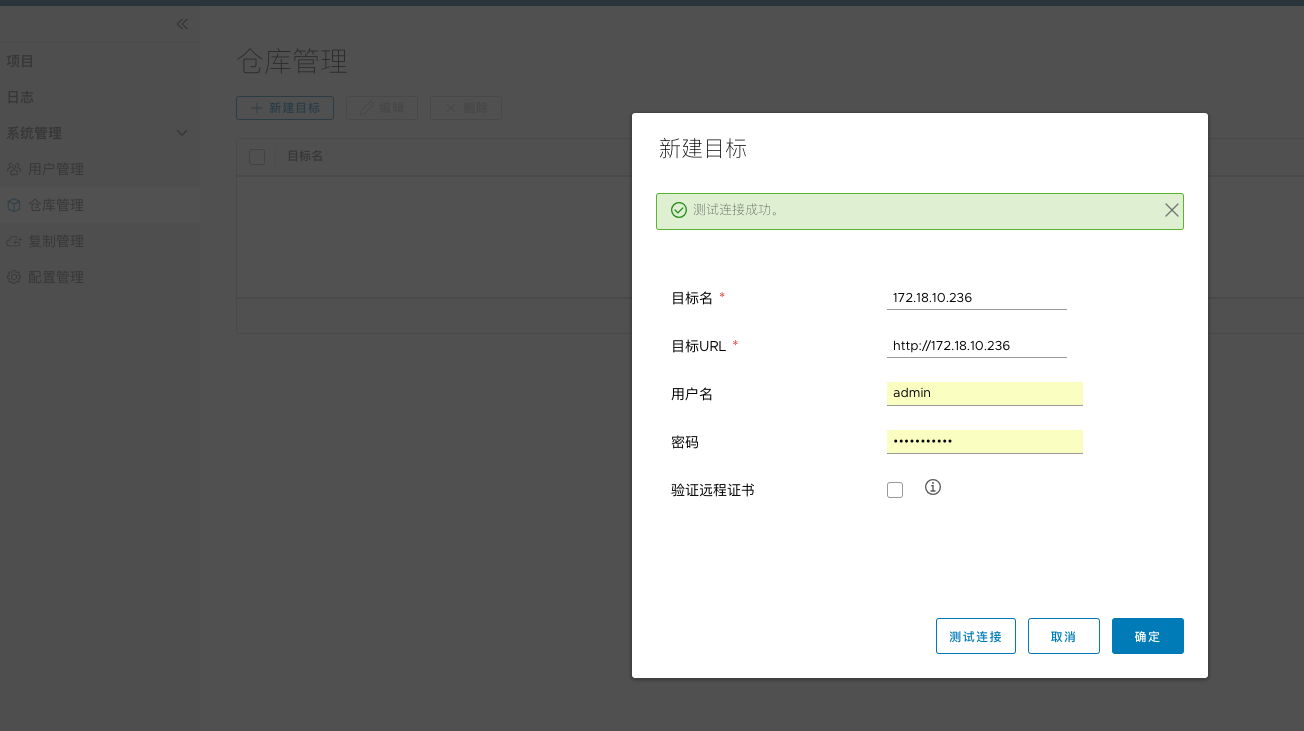
删除所有之前的数据：

rm -rf /data/\*

重新install一波：

./install.sh

在harbor主节点上配置复制镜像到该节点：





然后看一下节点1上已经同步了所有镜像：

